# MeshyDB Documentation

Yetisoftworks

Aug 11, 2019

# Getting Started


1 What is Meshy? 1

2 Before you get started! 3

3 Next Steps 5

CHAPTER 1

# What is Meshy?

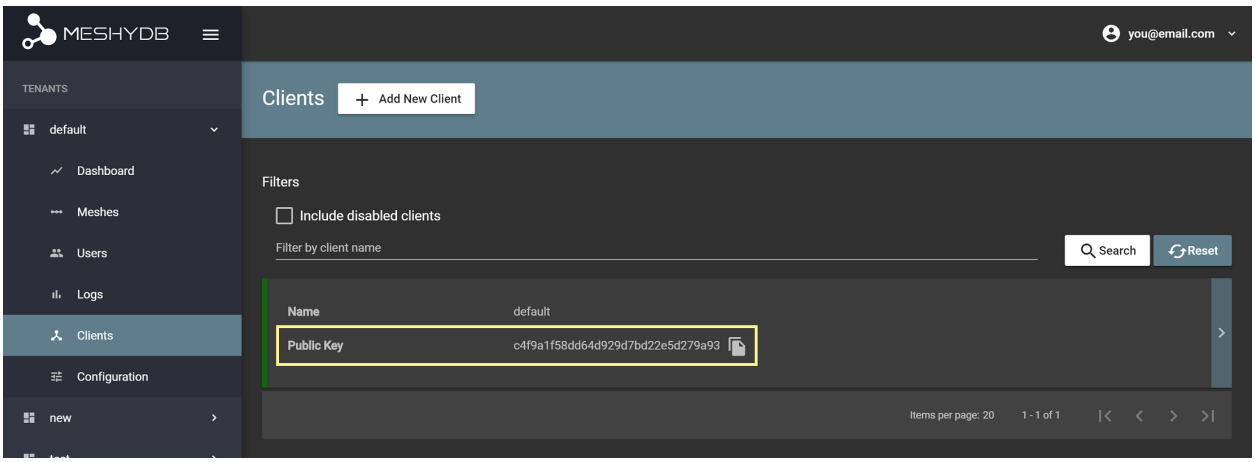
MeshyDB gives you a fully functional API backend in minutes. We take care of the bulky time-consuming API, letting you focus on the front-end design. Build apps faster by leveraging the MeshyDB backend.

CHAPTER 2

# Before you get started!

This documentation assumes you have an active MeshyDB account. If you do not, please create a free account at https://meshydb.com.

Once your account is verified, you will need to gather your from the Clients page under your default tenant. See image below:



# CHAPTER 3

# Next Steps

Now that you have your public key, you can begin with any of our language specific quick starts:

- C#
- NodeJS
- REST

## 3.1 C#

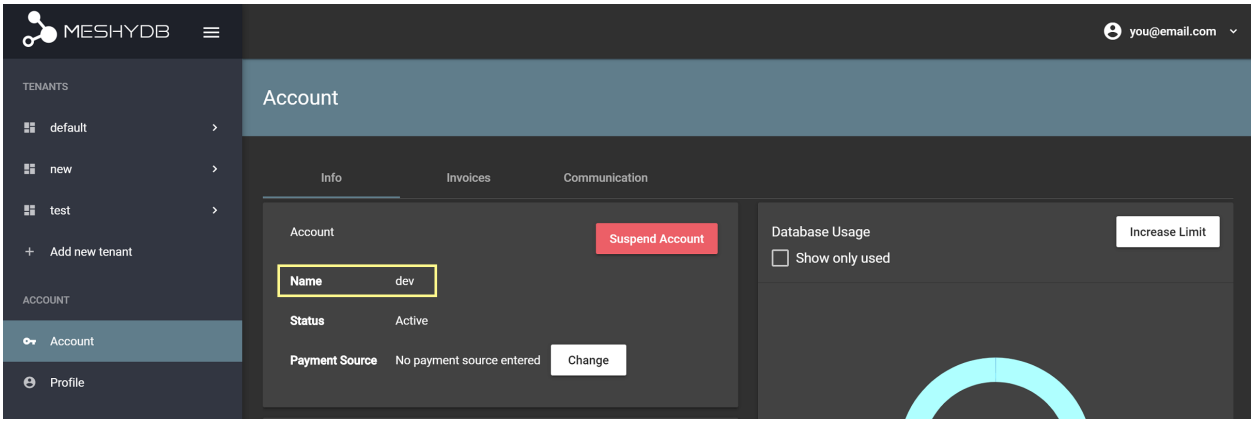
### 3.1.1 Before you get started!

This documentation assumes you have an active MeshyDB account. If you do not, please create a free account at https://meshydb.com.

Once you verify your account you will need to gather your and .

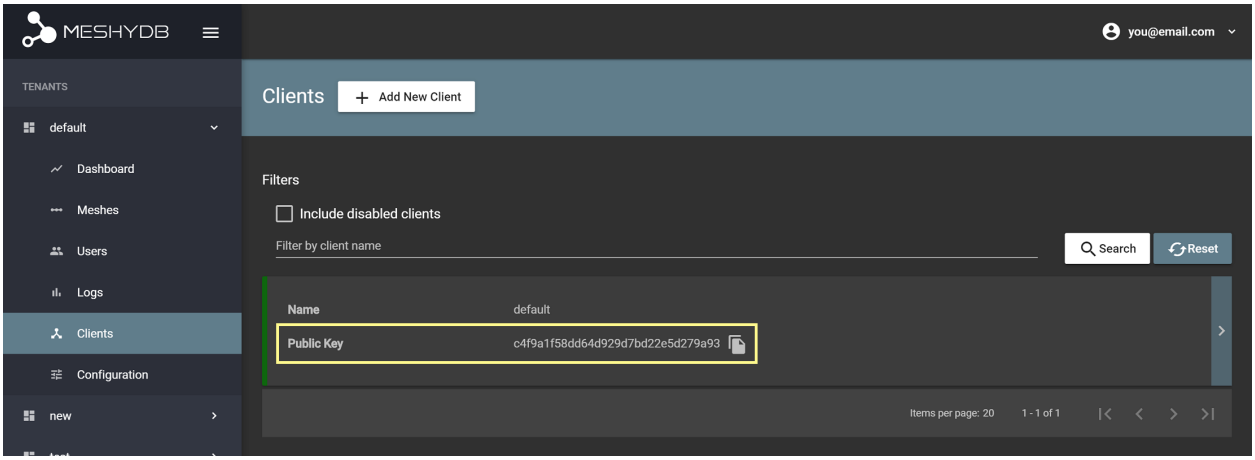
#### Identify Account Name

Your can be found under the Account page. See image below:




### Identify Public Key

Your can be found under the Clients page under your default tenant. See image below:




In the following we will assume no other configuration has been made to your account or tenants so we can just begin!

## 3.1.2 Install SDK

The supporting SDK is open source and you are able to use .Net Framework 4.7.1+ or .Net Core 2.x.

Let's install the MeshyDB.SDK NuGet package with the following command:

```
Install-Package MeshyDb.SDK
```

## 3.1.3 Initialize

The client is used to establish a connection to the API. You will need to provide your and from your account and client.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

### 3.1.4 Register Anonymous User

Anonymous users are great for associating data to people or devices without having them go through any type of user registration.

The example below shows verifying a username is available and registering an anonymous user if the username does not exist.

C#

```
var username = "mctesterton";

var userExists = await client.CheckUserExistAsync(username);

if (!userExists.Exists)
{
   await client.RegisterAnonymousUserAsync(username);
}
```

**username** [string] Unique identifier for user or device. If it is not provided a username will be automatically generated.

#### Responses

**201** [Created]

- New user has been registered and is now available for use.

Example Result

```
{
   "id": "5c78cc81dd870827a8e7b6c4",
   "username": "mctesterton",
   "firstName": null,
   "lastName": null,
   "verified": false,
   "isActive": true,
   "phoneNumber": null,
   "emailAddress": null,
   "roles": [],
   "securityQuestions": [],
   "anonymous": true
}
```

**400** [Bad request]

- Username is a required field.
- Anonymous registration is not enabled.
- Username must be unique.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

### 3.1.5 Login

All data interaction must be done on behalf of a user. This is done to ensure proper authorized access of your data.

The example below shows logging in an anonymous user.

C#

```
var connection = await client.LoginAnonymouslyAsync(username);
```

**username** [string, required] Unique identifier for user or device.

### Responses

**200** [OK]

- Generates new credentials for authorized user.

Example Result

```
{
  "access_token": "ey...",
  "expires_in": 3600,
  "token_type": "Bearer",
  "refresh_token": "ab23cd3343e9328g"
}
```

**400** [Bad request]

- Token is invalid.
- Client id is invalid.
- Grant type is invalid.
- User is no longer active.
- Invalid Scope.
- Username is invalid.
- Password is invalid.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

Once we login we can access our connection through a static member.

C#

```
connection = MeshyClient.CurrentConnection;
```

## 3.1.6 Retrieving Self

When a user is created they have some profile information that helps identify them. We can use this information to link their id back to data that has been created.

The example below shows retrieving information of the user.

C#

```
var user = await connection.Users.GetSelfAsync();
```

No parameters provided.

### Responses

**200** [OK]

- Retrieves information about the authorized user.

Example Result

```
{
   "id": "5c78cc81dd870827a8e7b6c4",
   "username": "mctesterton",
   "firstName": null,
   "lastName": null,
   "verified": false,
   "isActive": true,
   "phoneNumber": null,
   "emailAddress": null,
   "roles": [],
   "securityQuestions": [],
   "anonymous": true
}
```

**401** [Unauthorized]

- User is not authorized to make call.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

## 3.1.7 Create data

Now that a user connection is established you can begin making API requests.

The MeshyDB SDK requires all data extend the class.

The example below shows a Person represented by a first name, last name and user id.

C#

```
// Mesh Name can be overridden by attribute, otherwise by default it is derived from␣
↪class name
[MeshName("Person")]
public class Person : MeshData
{
   public string FirstName { get; set; }
   public string LastName { get; set; }
   public string UserId { get; set; }
}
```

Now that we have a representation of a person we can start making data to write to the API.

The example below shows committing a new person.

C#

```
var model = new Person()
{
   FirstName = "Bob",
   LastName = "Bobson",
```

(continues on next page)

(continued from previous page)

```
   UserId = user.Id
};

model = await connection.Meshes.CreateAsync(model);
```

**model** [object, required] Representation of data that *must* extend .

#### Responses

**201** [Created]

- Result of newly created mesh data.

Example Result

```
{
   "_id":"5d438ff23b0b7dd957a765ce",
   "firstName": "Bob",
   "lastName": "Bobson",
   "userId": "5c78cc81dd870827a8e7b6c4"
}
```

**400** [Bad request]

- Mesh name is invalid and must be alpha characters only.
- Mesh property cannot begin with '$' or contain '.'.

**401** [Unauthorized]

- User is not authorized to make call.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

### 3.1.8 Update data

The API allows you to make updates to specific by targeting the id.

The SDK makes this even simpler since the id can be derived from the object itself along with all it's modifications.

The example below shows modifying the first name and committing those changes to the API.

C#

```
model.FirstName = "Robert";

model = await connection.Meshes.UpdateAsync(model);
```

**model** [object, required] Representation of data that *must* extend .

#### Responses

**200** [OK]

- Result of updated mesh data.

Example Result

```
{
   "_id":"5d438ff23b0b7dd957a765ce",
   "firstName": "Robert",
   "lastName": "Bobson",
   "userId": "5c78cc81dd870827a8e7b6c4"
}
```

**400** [Bad request]

- Mesh name is invalid and must be alpha characters only.
- Mesh property cannot begin with '$' or contain '.'.

**401** [Unauthorized]

- User is not authorized to make call.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

### 3.1.9 Search data

The API allows you to search using a Linq expression.

The example below shows searching based where the first name starts with Rob.

C#

```
Expression<Func<Person, bool>> filter = (Person x) => x.FirstName.StartsWith("Rob");

var pagedPersonResult = await connection.Meshes
                                        .SearchAsync<Person>(filter);
```

**filter** [object] Criteria provided in a Linq expression to limit results.

#### Responses

**200** [OK]

- Mesh data found with given search criteria.

Example Result

```
{
   "page": 1,
   "pageSize": 25,
   "results":  [{
                  "_id":"5d438ff23b0b7dd957a765ce",
                  "firstName": "Robert",
                  "lastName": "Bobson",
                  "userId": "5c78cc81dd870827a8e7b6c4"
               }],
   "totalRecords": 1
}
```

**400** [Bad request]

- Mesh name is invalid and must be alpha characters only.
- Filter is in an invalid format. It must be in a valid Mongo DB format.
- Order by is in an invalid format. It must be in a valid Mongo DB format.

**401** [Unauthorized]

- User is not authorized to make call.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

In some cases you may need more control on your filtering or sorting. You can optionally provide this criteria in a MongoDB format.

### 3.1.10 Delete data

The API allows you to delete a specific by targeting the id.

The example below shows deleting the data from the API by providing the object.

*Deleted* data is not able to be recovered. If you anticipate the need to recover this data please implementing a .

C#

```
var id = model.Id;

await connection.Meshes.DeleteAsync<Person>(id);
```

**id** [string, required] Identifier of record that must be deleted.

#### Responses

**204** [No Content]

- Mesh has been deleted successfully.

**400** [Bad request]

- Mesh name is invalid and must be alpha characters only.

**401** [Unauthorized]

- User is not authorized to make call.

**404** [Not Found]

- Mesh data was not found.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

### 3.1.11 Sign out

The MeshyDB SDK manages a single connection to the API.

The Meshy SDK handles token management, this includes refresh tokens used to maintain a user's connection.

As a result it is recommended to implement Sign Out to ensure the current user is logged out and all refresh tokens are revoked.

The example below shows signing out of the currently established connection.

C#

```
await connection.SignoutAsync();
```

No parameters provided. The connection is aware of who needs to be signed out.

#### Responses

**200** [OK]

- Identifies successful logout.

**400** [Bad request]

- Invalid client id.
- Token is missing.
- Unsupported Token type.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

Not seeing something you need? Feel free to give us a chat or contact us at support@meshydb.com.

## 3.2 NodeJS

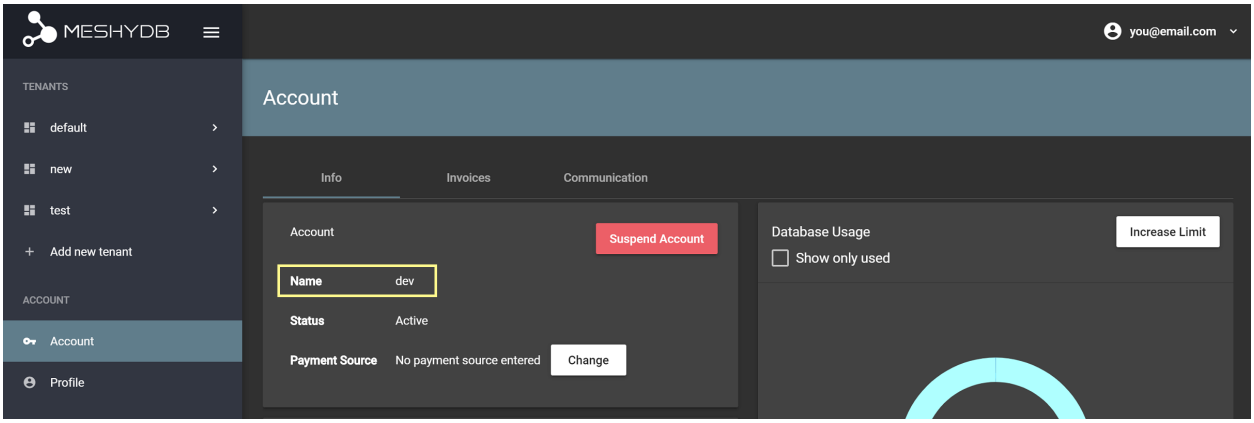
### 3.2.1 Before you get started!

This documentation assumes you have an active MeshyDB account. If you do not, please create a free account at https://meshydb.com.

Once you verify your account you will need to gather your and .

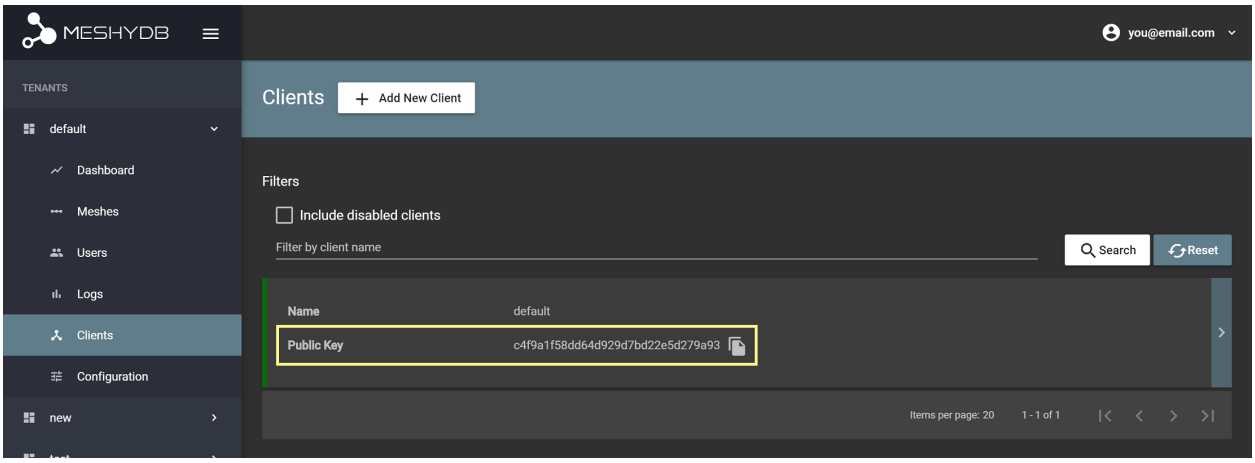
#### Identify Account Name

Your can be found under the Account page. See image below:



### Identify Public Key

Your can be found under the Clients page under your default tenant. See image below:




In the following we will assume no other configuration has been made to your account or tenants so we can just begin!

## 3.2.2 Install SDK

The supporting SDK is open source and you are able to use a browser or NodeJS application.

Let's install the MeshyDB.SDK NPM package with the following command:

```
npm install @meshydb/sdk
```

## 3.2.3 Initialize

The client is used to establish a connection to the API. You will need to provide your and from your account and client.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

## 3.2.4 Register Anonymous User

Anonymous users are great for associating data to people or devices without having them go through any type of user registration.

The example below shows verifying a username is available and registering an anonymous user if the username does not exist.

NodeJS

```
var username = "mctesterton";

var userExists = await client.checkUserExist(username);
```

(continues on next page)

(continued from previous page)

```
if (!userExists.exists) {
   await client.registerAnonymousUser(username);
}
```

**username** [string] Unique identifier for user or device. If it is not provided a username will be automatically generated.

### Responses

**201** [Created]

- New user has been registered and is now available for use.

Example Result

```
{
   "id": "5c78cc81dd870827a8e7b6c4",
   "username": "mctesterton",
   "firstName": null,
   "lastName": null,
   "verified": false,
   "isActive": true,
   "phoneNumber": null,
   "emailAddress": null,
   "roles": [],
   "securityQuestions": [],
   "anonymous": true
}
```

**400** [Bad request]

- Username is a required field.
- Anonymous registration is not enabled.
- Username must be unique.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

## 3.2.5 Login

All data interaction must be done on behalf of a user. This is done to ensure proper authorized access of your data.

The example below shows logging in an anonymous user.

NodeJS

```
var connection = await client.loginAnonymously(username);
```

**username** [string, required] Unique identifier for user or device.

### Responses

**200** [OK]

- Generates new credentials for authorized user.

Example Result

```
{
  "access_token": "ey...",
  "expires_in": 3600,
  "token_type": "Bearer",
  "refresh_token": "ab23cd3343e9328g"
}
```

**400** [Bad request]

- Token is invalid.
- Client id is invalid.
- Grant type is invalid.
- User is no longer active.
- Invalid Scope.
- Username is invalid.
- Password is invalid.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

Once we login we can access our connection through a static member.

NodeJS

```
connection = MeshyClient.currentConnection;
```

## 3.2.6 Retrieving Self

When a user is created they have some profile information that helps identify them. We can use this information to link their id back to data that has been created.

The example below shows retrieving information of the user.

NodeJS

```
var user = await connection.usersService.getSelf();
```

No parameters provided.

### Responses

**200** [OK]

- Retrieves information about the authorized user.

Example Result

```
{
   "id": "5c78cc81dd870827a8e7b6c4",
   "username": "mctesterton",
   "firstName": null,
   "lastName": null,
   "verified": false,
   "isActive": true,
   "phoneNumber": null,
   "emailAddress": null,
   "roles": [],
   "securityQuestions": [],
   "anonymous": true
}
```

**401** [Unauthorized]

- User is not authorized to make call.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

### 3.2.7 Create data

Now that a user connection is established you can begin making API requests.

The example below shows committing a new person.

NodeJS

```
var model = {
                _id: undefined,
                firstName: "Bob",
                lastName: "Bobson",
                userId: user.id
            };

var meshName = "person";

model = await connection.meshesService.create(meshName, model);
```

**meshName** [string, required] Identifies which mesh collection to manage.

**model** [object, required] Represents a person in this example.

#### Responses

**201** [Created]

- Result of newly created mesh data.

Example Result

```
{
   "_id":"5d438ff23b0b7dd957a765ce",
   "firstName": "Bob",
   "lastName": "Bobson",
```

(continues on next page)

(continued from previous page)

```
    "userId": "5c78cc81dd870827a8e7b6c4"
}
```

**400** [Bad request]

- Mesh name is invalid and must be alpha characters only.
- Mesh property cannot begin with '$' or contain '.'.

**401** [Unauthorized]

- User is not authorized to make call.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

### 3.2.8 Update data

The API allows you to make updates to specific Mesh Data by targeting the id.

The SDK makes this even simpler since the id can be derived from the object itself along with all it's modifications.

The example below shows modifying the first name and committing those changes to the API.

NodeJS

```
model.firstName = "Robert";

model = await connection.meshesService.update(meshName, model);
```

**meshName** [string, required] Identifies which mesh collection to manage.

**model** [object, required] Represents a person in this example.

#### Responses

**200** [OK]

- Result of updated mesh data.

Example Result

```
{
    "_id":"5d438ff23b0b7dd957a765ce",
    "firstName": "Robert",
    "lastName": "Bobson",
    "userId": "5c78cc81dd870827a8e7b6c4"
}
```

**400** [Bad request]

- Mesh name is invalid and must be alpha characters only.
- Mesh property cannot begin with '$' or contain '.'.

**401** [Unauthorized]

- User is not authorized to make call.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

### 3.2.9 Search data

The API allows you to search a mesh collection using a MongoDB expression.

The example below shows searching based where the first name starts with Rob.

NodeJS

```
var filter = { 'firstName': { "$regex": "^Rob" } };

var pagedPersonResult = await connection.meshesService
                                        .search(meshName, { filter: filter });
```

**meshName** [string, required] Identifies which mesh collection to manage.

**filter** [object] Criteria provided in a MongoDB expression to limit results.

#### Responses

**200** [OK]

- Mesh data found with given search criteria.

Example Result

```
{
   "page": 1,
   "pageSize": 25,
   "results":  [{
                  "_id":"5d438ff23b0b7dd957a765ce",
                  "firstName": "Robert",
                  "lastName": "Bobson",
                  "userId": "5c78cc81dd870827a8e7b6c4"
               }],
   "totalRecords": 1
}
```

**400** [Bad request]

- Mesh name is invalid and must be alpha characters only.
- Filter is in an invalid format. It must be in a valid Mongo DB format.
- Order by is in an invalid format. It must be in a valid Mongo DB format.

**401** [Unauthorized]

- User is not authorized to make call.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

### 3.2.10 Delete data

The API allows you to delete a specific Mesh Data by targeting the id.

The example below shows deleting the data from the API by providing the object.

*Deleted* data is not able to be recovered. If you anticipate the need to recover this data please implementing a .

NodeJS

```
var id = model._id;

await connection.meshesService.delete(meshName, id);
```

**meshName** [string, required] Identifies which mesh collection to manage.

**id** [string, required] Identifier of record that must be deleted.

#### Responses

**204** [No Content]

- Mesh has been deleted successfully.

**400** [Bad request]

- Mesh name is invalid and must be alpha characters only.

**401** [Unauthorized]

- User is not authorized to make call.

**404** [Not Found]

- Mesh data was not found.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

### 3.2.11 Sign out

The MeshyDB SDK manages a single connection to the API.

The Meshy SDK handles token management, this includes refresh tokens used to maintain a user's connection.

As a result it is recommended to implement Sign Out to ensure the current user is logged out and all refresh tokens are revoked.

The example below shows signing out of the currently established connection.

NodeJS

```
await connection.signout();
```

No parameters provided. The connection is aware of who needs to be signed out.

#### Responses

**200** [OK]

- Identifies successful logout.

**400** [Bad request]

- Invalid client id.
- Token is missing.

- Unsupported Token type.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

Not seeing something you need? Feel free to give us a chat or contact us at support@meshydb.com.

## 3.3 REST

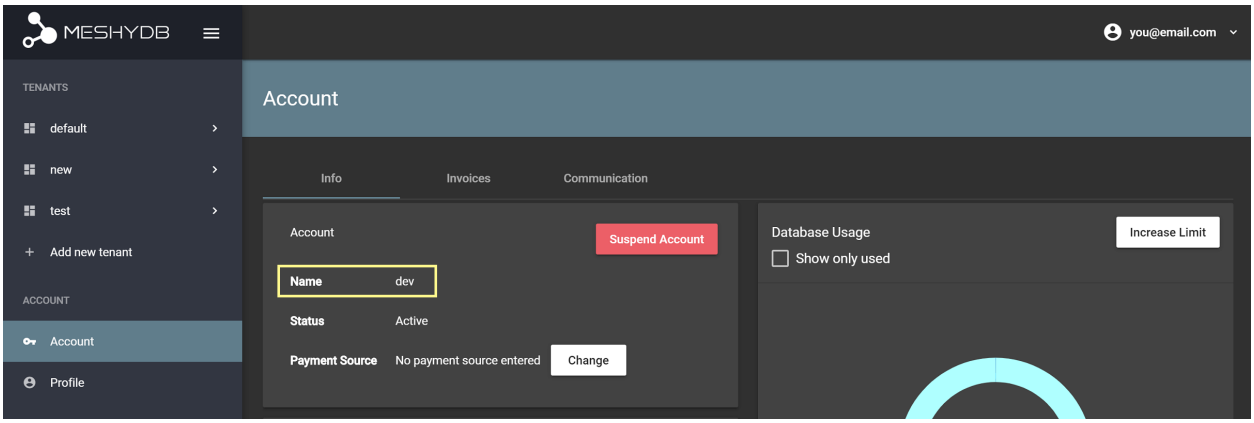
### 3.3.1 Before you get started!

This documentation assumes you have an active MeshyDB account. If you do not, please create a free account at https://meshydb.com.

Once you verify your account you will need to gather your and .

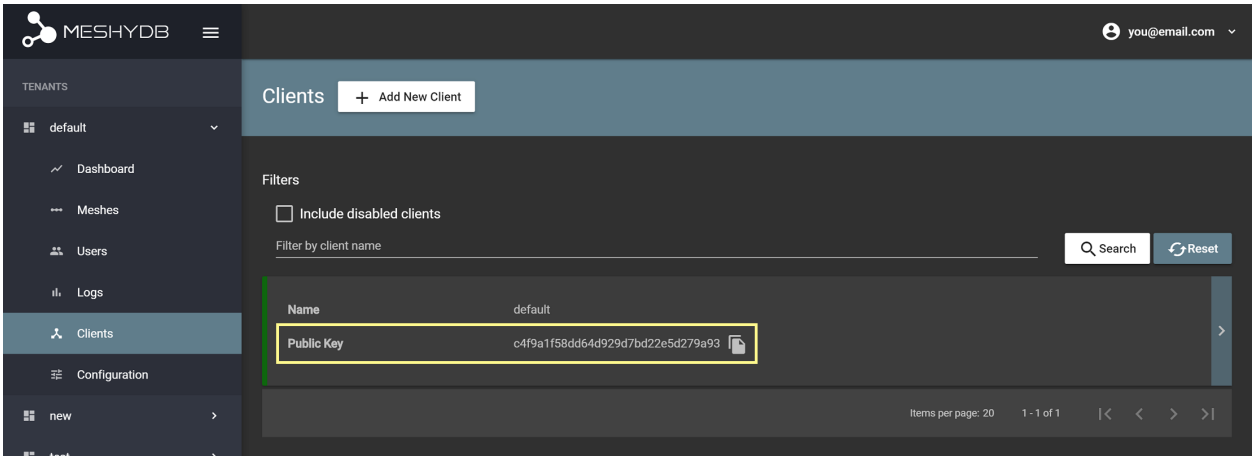
#### Identify Account Name

Your can be found under the Account page. See image below:




#### Identify Public Key

Your can be found under the Clients page under your default tenant. See image below:



In the following we will assume no other configuration has been made to your account or tenants so we can just begin!

### 3.3.2 Checking Username Exists

Checking username helps identify if a device or user has already registered.

The example below shows verifying a username is available.

REST

```
GET https://api.meshydb.com/{accountName}/users/{username}/exists HTTP/1.1
```

**accountName** [string, required] Indicates which account you are connecting to.

**username** [string, required] Unique identifier for user or device.

#### Responses

**201** [Created]

- Identifies if username already exists.

Example Result

```
{
   "exists": false
}
```

**400** [Bad request]

- Username is required.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

### 3.3.3 Register Anonymous User

Anonymous users are great for associating data to people or devices without having them go through any type of user registration.

The example below shows registering an anonymous user.

REST

```
POST https://api.meshydb.com/{accountName}/users/register/anonymous HTTP/1.1
Content-Type: application/json

  {
    "username": "mctesterton"
  }
```

**accountName** [string, required] Indicates which account you are connecting to.

**username** [string, required] Unique identifier for user or device.

### Responses

**201** [Created]

- New user has been registered and is now available for use.

Example Result

```
{
   "id": "5c78cc81dd870827a8e7b6c4",
   "username": "mctesterton",
   "firstName": null,
   "lastName": null,
   "verified": false,
   "isActive": true,
   "phoneNumber": null,
   "emailAddress": null,
   "roles": [],
   "securityQuestions": [],
   "anonymous": true
}
```

**400** [Bad request]

- Username is a required field.
- Anonymous registration is not enabled.
- Username must be unique.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

## 3.3.4 Login

All data interaction must be done on behalf of a user. This is done to ensure proper authorized access of your data.

The example below shows logging in an anonymous user.

REST

```
POST https://auth.meshydb.com/{accountName}/connect/token HTTP/1.1
Content-Type: application/x-www-form-urlencoded

   client_id={publicKey}&
   grant_type=password&
   username={username}&
   password=nopassword&
   scope=meshy.api offline_access
```

(Form-encoding removed, and line breaks added for readability)

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**username** [string, required] Unique identifier for user or device.

**password** [string, required] User secret credentials for login. When anonymous it is static as nopassword.

### Responses

**200** [OK]

- Generates new credentials for authorized user. The token will expire and will need to be refreshed.

Example Result

```
{
  "access_token": "ey...",
  "expires_in": 3600,
  "token_type": "Bearer",
  "refresh_token": "ab23cd3343e9328g"
}
```

**400** [Bad request]

- Token is invalid.
- Client id is invalid.
- Grant type is invalid.
- User is no longer active.
- Invalid Scope.
- Username is invalid.
- Password is invalid.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

## 3.3.5 Retrieving Self

When a user is created they have some profile information that helps identify them. We can use this information to link their id back to data that has been created.

The example below shows retrieving information of the user.

REST

```
GET https://api.meshydb.com/{accountName}/users/me HTTP/1.1
Authentication: Bearer {access_token}
```

**accountName** [string, required] Indicates which account you are connecting to.

**access_token** [string, required] Token identifying authorization with MeshyDB requested during Generating Token.

### Responses

**200** [OK]

- Retrieves information about the authorized user.

Example Result

```
{
   "id": "5c78cc81dd870827a8e7b6c4",
   "username": "mctesterton",
   "firstName": null,
   "lastName": null,
   "verified": false,
   "isActive": true,
   "phoneNumber": null,
   "emailAddress": null,
   "roles": [],
   "securityQuestions": [],
   "anonymous": true
}
```

**401** [Unauthorized]

- User is not authorized to make call.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

### 3.3.6 Create data

Now that a user is authorized you can begin making API requests.

The example below shows committing a new such as a person.

REST

```
POST https://api.meshydb.com/{accountName}/meshes/{meshName} HTTP/1.1
Authentication: Bearer {access_token}
Content-Type: application/json

   {
      "firstName": "Bob",
      "lastName": "Bobson",
      "userId": "5c78cc81dd870827a8e7b6c4"
   }
```

**accountName** [string, required] Indicates which account you are connecting to.

**access_token** [string, required] Token identifying authorization with MeshyDB requested during *Login*.

**meshName** [string, required] Identifies name of mesh collection. e.g. person.

#### Responses

**201** [Created]

- Result of newly created mesh data.

Example Result

```
{
   "_id":"5d438ff23b0b7dd957a765ce",
   "firstName": "Bob",
   "lastName": "Bobson",
```

(continues on next page)

(continued from previous page)

```
    "userId": "5c78cc81dd870827a8e7b6c4"
}
```

**400** [Bad request]

- Mesh name is invalid and must be alpha characters only.
- Mesh property cannot begin with '$' or contain '.'.

**401** [Unauthorized]

- User is not authorized to make call.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

### 3.3.7 Update data

The API allows you to make updates to specific by targeting the id.

The example below shows modifying the first name and committing those changes to the API.

REST

```
PUT https://api.meshydb.com/{accountName}/meshes/{meshName}/{id}  HTTP/1.1
Authentication: Bearer {access_token}
Content-Type: application/json

    {
        "firstName": "Robert",
        "lastName": "Bobson"
    }
```

**accountName** [string, required] Indicates which account you are connecting to.

**access_token** [string, required] Token identifying authorization with MeshyDB requested during *Login*.

**meshName** [string, required] Identifies name of mesh collection. e.g. person.

**id** [string, required] Identifies unique record of Mesh data to replace.

#### Responses

**200** [OK]

- Result of updated mesh data.

Example Result

```
{
    "_id":"5d438ff23b0b7dd957a765ce",
    "firstName": "Robert",
    "lastName": "Bobson",
    "userId": "5c78cc81dd870827a8e7b6c4"
}
```

**400** [Bad request]

- Mesh name is invalid and must be alpha characters only.

- Mesh property cannot begin with '$' or contain '.'.

**401** [Unauthorized]

- User is not authorized to make call.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

### 3.3.8 Search data

The API allows you to search using a MongoDB expression.

The example below shows searching based where the first name starts with Rob.

REST

```
GET https://api.meshydb.com/{accountName}/meshes/{meshName}?filter={ 'firstName': { "
↪$regex": "^Rob" } } HTTP/1.1
Authentication: Bearer {access_token}
```

(Encoding removed for readability)

**accountName** [string, required] Indicates which account you are connecting to.

**access_token** [string, required] Token identifying authorization with MeshyDB requested during *Login*.

**meshName** [string, required] Identifies name of mesh collection. e.g. person.

**filter** [string] Criteria provided in a MongoDB format to limit results.

**orderby** [string] Defines which fields need to be sorted and direction in a MongoDB format.

**page** [integer, default: 1] Page number of results to bring back.

**pageSize** [integer, max: 200, default: 25] Number of results to bring back per page.

#### Responses

**200** [OK]

- Mesh data found with given search criteria.

Example Result

```
{
   "page": 1,
   "pageSize": 25,
   "results":  [{
                  "_id":"5d438ff23b0b7dd957a765ce",
                  "firstName": "Robert",
                  "lastName": "Bobson",
                  "userId": "5c78cc81dd870827a8e7b6c4"
               }],
   "totalRecords": 1
}
```

**400** [Bad request]

- Mesh name is invalid and must be alpha characters only.

- Filter is in an invalid format. It must be in a valid Mongo DB format.
- Order by is in an invalid format. It must be in a valid Mongo DB format.

**401** [Unauthorized]

- User is not authorized to make call.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

### 3.3.9 Delete data

The API allows you to delete a specific by targeting the id.

The example below shows deleting the data from the API by providing the object.

*Deleted* data is not able to be recovered. If you anticipate the need to recover this data please implementing a .

REST

```
DELETE https://api.meshydb.com/{accountName}/meshes/{meshName}/{id} HTTP/1.1
Authentication: Bearer {access_token}
```

**accountName** [string, required] Indicates which account you are connecting to.

**access_token** [string, required] Token identifying authorization with MeshyDB requested during *Login*.

**meshName** [string, required] Identifies name of mesh collection. e.g. person.

**id** [string, required] Identifies unique record of Mesh data to remove.

#### Responses

**204** [No Content]

- Mesh has been deleted successfully.

**400** [Bad request]

- Mesh name is invalid and must be alpha characters only.

**401** [Unauthorized]

- User is not authorized to make call.

**404** [Not Found]

- Mesh data was not found.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

### 3.3.10 Sign out

When a user is authenticated a refresh token is generated. The refresh token allows a user to be silently authenticated.

As a result it is recommended to implement Sign Out to ensure the current user is logged out and all refresh tokens are revoked.

The example below shows revoking the refresh token. The access token is short lived and will expire within an hour.

REST

```
POST https://auth.meshydb.com/{accountName}/connect/revocation HTTP/1.1
Content-Type: application/x-www-form-urlencoded

  client_id={accountName}&
  grant_type=refresh_token&
  token={refresh_token}
```

(Line breaks added for readability)

**accountName** [string, required] Indicates which account you are connecting to.

**refresh_token** [string, required] Token to allow reauthorization with MeshyDB after the access token expires requested during *Login*.

### Responses

**200** [OK]

- Identifies successful logout.

**400** [Bad request]

- Invalid client id.
- Token is missing.
- Unsupported Token type.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

Not seeing something you need? Feel free to give us a chat or contact us at support@meshydb.com.

## 3.4 Generating Token

Create a short lived access token to be used for authorized API calls. Typically a token will last 3600 seconds(one hour).

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
var connection = client.LoginWithPassword(username, password);
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**username** [string, required] Unique identifier for user or device.

**password** [string, required] User secret credentials for login. When anonymous it is static as nopassword.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var meshyConnection = await client.login(username,password);
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**username** [string, required] Unique identifier for user or device.

**password** [string, required] User secret credentials for login. When anonymous it is static as nopassword.

REST

```
POST https://auth.meshydb.com/{accountName}/connect/token HTTP/1.1
Content-Type: application/x-www-form-urlencoded

   client_id={publicKey}&
   grant_type=password&
   username={username}&
   password={password}&
   scope=meshy.api offline_access
```

(Form-encoding removed, and line breaks added for readability)

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**username** [string, required] Unique identifier for user or device.

**password** [string, required] User secret credentials for login. When anonymous it is static as nopassword.

### Responses

**200** [OK]

- Generates new credentials for authorized user.

Example Result

```
{
   "access_token": "ey...",
   "expires_in": 3600,
   "token_type": "Bearer",
   "refresh_token": "ab23cd3343e9328g"
}
```

**400** [Bad request]

- Token is invalid.
- Client id is invalid.
- Grant type is invalid.
- User is no longer active.
- Invalid Scope.
- Username is invalid.
- Password is invalid.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

## 3.5 Refreshing Token

Using the token request made to generate an access token, a refresh token will also be generated.

Once the token expires the refresh token can be used to generate a new set of credentials for authorized calls.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
var connection = client.LoginWithPassword(username, password);
var refreshToken = connection.RetrieveRefreshToken();

connection = await client.LoginWithRefreshAsync(refreshToken);
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**username** [string, required] Unique identifier for user or device.

**password** [string, required] User secret credentials for login. When anonymous it is static as nopassword.

**refreshToken** [string, required] Refresh token generated from previous access token generation.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var meshyConnection = await client.login(username,password);

var refreshToken = meshyConnection.retrieveRefreshToken();

var refreshedMeshyConnection = await client.loginWithRefresh(refreshToken);
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**username** [string, required] Unique identifier for user or device.

**password** [string, required] User secret credentials for login. When anonymous it is static as nopassword.

**refreshToken** [string, required] Refresh token generated from previous access token generation.

REST

```
POST https://auth.meshydb.com/{accountName}/connect/token HTTP/1.1
Content-Type: application/x-www-form-urlencoded

   client_id={publicKey}&
   grant_type=refresh_token&
   refresh_token={refresh_token}
```

(Form-encoding removed, and line breaks added for readability)

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**refresh_token** [string, required] Refresh token generated from previous access token generation.

### Responses

**200** [OK]

- Generates new refresh credentials for authorized user.

Example Result

```
{
   "access_token": "ey...",
   "expires_in": 3600,
   "token_type": "Bearer",
   "refresh_token": "ab23cd3343e9328g"
}
```

**400** [Bad request]

- Token is invalid.
- Client id is invalid.
- Grant type is invalid.
- User is no longer active.
- Refresh token is expired.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

## 3.6 Creating Users

A user can be authenticated with the system for ensuring they are authorized to interact with the system.

You can either generate an anonymous user, or device user with limited functionality. Otherwise you can register a new user with full credentials.

### 3.6.1 Checking Username Available

Before identifying a device or unique user you can check to see if they already were registered.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);

var userExists = await client.CheckUserExistAsync(username);
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**username** [string, required] Unique identifier for user or device.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var userExists = await client.checkUserExist(username);
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**username** [string, required] Unique identifier for user or device.

REST

```
GET https://api.meshydb.com/{accountName}/users/{username}/exists HTTP/1.1
```

**accountName** [string, required] Indicates which account you are connecting to.

**username** [string, required] Unique identifier for user or device.

#### Responses

**201** [Created]

- Identifies if username already exists.

Example Result

```
{
   "exists": false
}
```

**400** [Bad request]

- Username is required.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

### 3.6.2 Registering Anonymous User

An anonymous user can identify a device or unique user without requiring user interaction.

This kind of user has limited functionality such as not having the ability to be verified or be assigned roles.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);

var anonymousUser = await client.RegisterAnonymousUserAsync(userName);
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**username** [string, required] Unique identifier for user or device.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var anonymousUser = await client.registerAnonymousUser(username);
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**username** [string, required] Unique identifier for user or device.

REST

```
POST https://api.meshydb.com/{accountName}/users/register/anonymous HTTP/1.1
Content-Type: application/json

  {
    "username": "username_testermctesterson"
  }
```

**accountName** [string, required] Indicates which account you are connecting to.

**username** [string, required] Unique identifier for user or device.

### Responses

**201** [Created]

- New user has been registered and is now available for use.

Example Result

```
{
   "id": "5c78cc81dd870827a8e7b6c4",
   "username": "username_testermctesterson",
   "firstName": null,
   "lastName": null,
   "verified": false,
   "isActive": true,
   "phoneNumber": null,
   "emailAddress": null,
   "roles": [],
   "securityQuestions": [],
   "anonymous": true
}
```

**400** [Bad request]

- Username is a required field.
- Anonymous registration is not enabled.
- Username must be unique.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

## 3.6.3 Registering User

Registering a user allows user defined credentials to access the system.

If email or text verification is configured, they will be prompted to verify their account.

The user will not be able to be authenticated until verification has been completed. The verification request lasts one hour before it expires.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);

var user = new RegisterUser();

await client.RegisterUserAsync(user);
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**username** [string, required] Unique identifier for user or device.

**newPassword** [string, required] New user secret credentials for login.

**firstName** [string] First name of registering user.

**lastName** [string] Last name of registering user.

**phoneNumber** [string, required *if using phone verification*] Phone number of registering user.

**emailAddress** [string, required *if using email verification*] Email address of registering user.

**securityQuestions** [object[], required *if using question verification*] New set of questions and answers for registering user in password recovery.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var user = await client.registerUser({
                                        username: username,
                                        newPassword: newPassword,
                                        firstName: firstName,
                                        lastName: lastName,
                                        phoneNumber: phoneNumber,
                                        emailAddress: emailAddress,
                                        securityQuestions: securityQuestions
                                    });
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**username** [string, required] Unique identifier for user or device.

**newPassword** [string, required] New user secret credentials for login.

**firstName** [string] First name of registering user.

**lastName** [string] Last name of registering user.

**phoneNumber** [string, required *if using phone verification*] Phone number of registering user.

**emailAddress** [string, required *if using email verification*] Email address of registering user.

**securityQuestions** [object[], required *if using question verification*] New set of questions and answers for registering user in password recovery.

REST

```
POST https://api.meshydb.com/{accountName}/users/register HTTP/1.1
Content-Type: application/json

  {
```

(continues on next page)

(continued from previous page)

```
    "username": "username_testermctesterson",
    "firstName": "Tester",
    "lastName": "McTesterton",
    "phoneNumber": "+15555555555",
    "emailAddress": "test@test.com",
    "securityQuestions": [
                            {
                               "question": "What would you say to this question?",
                               "answer": "mceasy123"
                            }
                         ],
    "newPassword": "newPassword"
  }
```

**accountName** [string, required] Indicates which account you are connecting to.

**username** [string, required] Unique identifier for user or device.

**newPassword** [string, required] New user secret credentials for login.

**firstName** [string] First name of registering user.

**lastName** [string] Last name of registering user.

**phoneNumber** [string, required *if using phone verification*] Phone number of registering user.

**emailAddress** [string, required *if using email verification*] Email address of registering user.

**securityQuestions** [object[], required *if using question verification*] New set of questions and answers for registering user in password recovery.

### Responses

**201** [Created]

- New user has been registered and must be verified before use.

Example Result

```
{
   "username": "username_testermctesterson",
   "attempt": 1,
   "hash": "...",
   "expires": "1/1/1900",
   "hint": "..."
}
```

**204** [No Content]

- New user has been registered and is now available for use.

**400** [Bad request]

- Public registration is not enabled.
- Email address is required when Email recovery is enabled.
- Phone number is required when Text recovery is enabled.
- At least one Security Questions is required when Question recovery is enabled.
- Username is a required field.

- Email address must be in a valid format.
- Phone number must be in an international format.
- Username must be unique.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

### 3.6.4 Check Hash

Optionally, before verifying the request you can choose to check if the verification code provided is valid.

You may want to provide this flow if you still need to collect more information about the user before finalizing verification.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);

var check = new UserVerificationCheck();

var isValid = await client.CheckHashAsync(check);
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**username** [string, required] Unique identifier for user or device.

**attempt** [integer, required] Identifies which attempt hash was generated against.

**hash** [string, required] Generated hash from verification request.

**expires** [date, required] Identifies when the request expires.

**hint** [string, required] Hint for verification code was generated.

**verificationCode** [string, required] Value to verify against verification request.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

await client.checkHash({
                            username: username,
                            attempt: attempt:
                            hash: hash,
                            expires: expires,
                            hint: hint,
                            verificationCode: verificationCode
                        });
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**username** [string, required] Unique identifier for user or device.

**attempt** [integer, required] Identifies which attempt hash was generated against.

**hash** [string, required] Generated hash from verification request.

**expires** [date, required] Identifies when the request expires.

**hint** [string, required] Hint for verification code was generated.

**verificationCode** [string, required] Value to verify against verification request.

REST

```
POST https://api.meshydb.com/{accountName}/users/checkhash HTTP/1.1
Content-Type: application/json

  {
     "username": "username_testermctesterson",
     "attempt": 1,
     "hash": "...",
     "expires": "1/1/1900",
     "hint": "...",
     "verificationCode": "...",
  }
```

**accountName** [string, required] Indicates which account you are connecting to.

**username** [string, required] Unique identifier for user or device.

**attempt** [integer, required] Identifies which attempt hash was generated against.

**hash** [string, required] Generated hash from verification request.

**expires** [date, required] Identifies when the request expires.

**hint** [string, required] Hint for verification code was generated.

**verificationCode** [string, required] Value to verify against verification request.

#### Responses

**200** [OK]

- Identifies if hash with verification code is valid.

Example Result

```
true
```

**400** [Bad request]

- Username is required.
- Hash is required.
- Expires is required.
- Verification code is required.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

### 3.6.5 Verify

If email or text verification is configured the registered user must be verified. The resulting request lasts one hour.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);

var check = new UserVerificationCheck();

await client.VerifyAsync(check);
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**username** [string, required] Unique identifier for user or device.

**attempt** [integer, required] Identifies which attempt hash was generated against.

**hash** [string, required] Generated hash from verification request.

**expires** [date, required] Identifies when the request expires.

**hint** [string, required] Hint for verification code was generated.

**verificationCode** [string, required] Value to verify against verification request.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

await client.verify({
                  username: username,
                  attempt: attempt:
                  hash: hash,
                  expires: expires,
                  hint: hint,
                  verificationCode: verificationCode
                                        });
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**username** [string, required] Unique identifier for user or device.

**attempt** [integer, required] Identifies which attempt hash was generated against.

**hash** [string, required] Generated hash from verification request.

**expires** [date, required] Identifies when the request expires.

**hint** [string, required] Hint for verification code was generated.

**verificationCode** [string, required] Value to verify against verification request.

REST

```
POST https://api.meshydb.com/{accountName}/users/verify HTTP/1.1
Content-Type: application/json

  {
     "username": "username_testermctesterson",
     "attempt": 1,
     "hash": "...",
     "expires": "1/1/1900",
     "hint": "...",
```

(continues on next page)

(continued from previous page)

```
    "verificationCode": "...",
}
```

**accountName** [string, required] Indicates which account you are connecting to.

**username** [string, required] Unique identifier for user or device.

**attempt** [integer, required] Identifies which attempt hash was generated against.

**hash** [string, required] Generated hash from verification request.

**expires** [date, required] Identifies when the request expires.

**hint** [string, required] Hint for verification code was generated.

**verificationCode** [string, required] Value to verify against verification request.

### Responses

**204** [No Content]

- User has been verified successfully.

**400** [Bad request]

- Username is required.
- Hash is required.
- Expires is required.
- Verification code is required.
- Hash is expired.
- Anonymous user cannot be verified.
- User has already been verified.
- Request hash is invalid.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

## 3.7 Retrieving Self

Retrieve details about the authenticated user.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginAnonymouslyAsync(username);

await connection.Users.GetLoggedInUserAsync();
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**username** [string, required] Unique identifier for user or device.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var anonymousUser = await client.registerAnonymousUser();

var meshyConnection = await client.loginAnonymously(anonymousUser.username);

var self = await meshyConnection.usersService.getSelf();
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**username** [string, required] Unique identifier for user or device.

REST

```
GET https://api.meshydb.com/{accountName}/users/me HTTP/1.1
Authentication: Bearer {access_token}
```

**accountName** [string, required] Indicates which account you are connecting to.

**access_token** [string, required] Token identifying authorization with MeshyDB requested during Generating Token.

### Responses

**200** [OK]

- Retrieves information about the authorized user.

Example Result

```
{
  "id": "5c78cc81dd870827a8e7b6c4",
  "username": "username_testermctesterson",
  "firstName": "Tester",
  "lastName": "McTesterton",
  "verified": true,
  "isActive": true,
  "phoneNumber": "5555555555",
  "roles": [
              "admin",
              "test"
           ],
  "securityQuestions": [
                          {
                             "question": "What would you say to this question?",
                             "answer": "..."
                          }
                       ],
  "anonymous": true
}
```

**401** [Unauthorized]

- User is not authorized to make call.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

## 3.8 Updating Self

Updating self allows the ability to update the authenticated user's information.

This might be personal or security questions for password recovery later.

### 3.8.1 Personal Information

The following can be used to update an authenticated user's personal information such as name, phone number, and email address.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginAnonymouslyAsync(username);

var user = new User();

await connection.Users.UpdateUserAsync(id, user);
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**firstName** [string] First name of authenticated user.

**lastName** [string] Last name of authenticated user.

**phoneNumber** [string, required *if using phone verification*] Phone number of authenticated user.

**emailAddress** [string, required *if using email verification*] Email address of authenticated user.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var anonymousUser = await client.registerAnonymousUser();

var meshyConnection = await client.loginAnonymously(anonymousUser.username);

var self = await meshyConnection.usersService.updateSelf({
                                                    firstName: firstName,
                                                    lastName: lastName,
                                                    phoneNumber: phoneNumber,
                                                    emailAddress: emailAddress
                                                });
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**firstName** [string] First name of authenticated user.

**lastName** [string] Last name of authenticated user.

**phoneNumber** [string, required *if using phone verification*] Phone number of authenticated user.

**emailAddress** [string, required *if using email verification*] Email address of authenticated user.

REST

```
PUT https://api.meshydb.com/{accountName}/users/me HTTP/1.1
Authentication: Bearer {access_token}
Content-Type: application/json

  {
    "firstName": "Tester",
    "lastName": "McTesterton",
    "phoneNumber": "+15555555555",
    "emailAddress": "test@test.com"
  }
```

**accountName** [string, required] Indicates which account you are connecting to.

**access_token** [string, required] Token identifying authorization with MeshyDB requested during Generating Token.

**firstName** [string] First name of authenticated user.

**lastName** [string] Last name of authenticated user.

**phoneNumber** [string, required *if using phone verification*] Phone number of authenticated user.

**emailAddress** [string, required *if using email verification*] Email address of authenticated user.

### Responses

**200** [OK] Updated information of updated authorized user.

Example Result

```
{
  "id": "5c78cc81dd870827a8e7b6c4",
  "username": "username_testermctesterson",
  "firstName": "Tester",
  "lastName": "McTesterton",
  "verified": true,
  "isActive": true,
  "phoneNumber": "+15555555555",
  "emailAddress": "test@test.com",
  "roles": [
              "admin",
              "test"
           ],
  "securityQuestions": [
                          {
                             "question": "What would you say to this question?",
                             "answer": "..."
                          }
                       ],
  "anonymous": false
}
```

**400** [Bad request]

- Email address is required when Email recovery is enabled and the user is not anonymous.
- Phone number is required when Text recovery is enabled and the user is not anonymous.
- Username is a required field.
- Email address must be in a valid format.

- Phone number must be in an international format.
- Unable to change user roles via API.

**401** [Unauthorized]

- User is not authorized to make call.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

### 3.8.2 Security Questions

The following can be used to change the authenticated user's security questions to be used for password recovery.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginAnonymouslyAsync(username);

var questions = new UserSecurityQuestionUpdate();

questions.SecurityQuestions.Add(new SecurityQuestion(){
                                                        Question = "What should␣
↪this be?",
                                                        Answer = "This seems like␣
↪an ok example"
                                                    };

await connection.Users.UpdateSecurityQuestions(id, user);
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**securityQuestions** [object[], required] New set of questions and answers for authenticated user in password recovery.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var meshyConnection = await client.login(username, password);

await meshyConnection.usersService.updateSecurityQuestion({
                                                            securityQuestions:␣
↪securityQuestions
                                                        });
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**securityQuestions** [object[], required] Collection of questions and answers used for password recovery if question security is configured.

REST

```
POST https://api.meshydb.com/{accountName}/users/me/questions HTTP/1.1
Authentication: Bearer {access_token}
Content-Type: application/json
```

(continues on next page)

(continued from previous page)

```
{
  "securityQuestions": [
                         {
                             "question": "What would you say to this question?",
                             "answer": "..."
                         }
                       ]
}
```

**accountName** [string, required] Indicates which account you are connecting to.

**access_token** [string, required] Token identifying authorization with MeshyDB requested during Generating Token.

**securityQuestions** [object[], required] New set of questions and answers for authenticated user in password recovery.

#### Responses

**204** [No Content]

- Updated information of updated authorized user.

**400** [Bad request]

- Unable to update security questions if question verification is not configured.
- Anonymous user cannot have security questions.
- At least one question is required.
- Question text is required.
- Answer is required.

**401** [Unauthorized]

- User is not authorized to make call.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

### 3.8.3 Changing Password

Allows the authenticated user to change their password.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginWithPasswordAsync(username, password);

await connection.UpdatePasswordAsync(previousPassword, newPassword);
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**username** [string, required] Unique identifier for user or device.

**password** [string, required] User secret credentials for login. When anonymous it is static as nopassword.

**previousPassword** [string, required] Previous user secret credentials for login.

**newPassword** [string, required] New user secret credentials for login.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var meshyConnection = await client.login(username, password);

await meshyConnection.updatePassword(previousPassword, newPassword);
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**username** [string, required] Unique identifier for user or device.

**password** [string, required] User secret credentials for login. When anonymous it is static as nopassword.

**previousPassword** [string, required] Previous user secret credentials for login.

**newPassword** [string, required] New user secret credentials for login.

REST

```
POST https://api.meshydb.com/{accountName}/users/me/password HTTP/1.1
Authentication: Bearer {access_token}
Content-Type: application/json

  {
    "newPassword": "newPassword",
    "previousPassword": "previousPassword"
  }
```

**accountName** [string, required] Indicates which account you are connecting to.

**access_token: string, required** Token identifying authorization with MeshyDB requested during Generate Access Token.

**previousPassword** [string, required] Previous user secret credentials for login.

**newPassword** [string, required] New user secret credentials for login.

### Responses

**204** [No Content]

- Identifies password was updated successfully.

**400** [Bad request]

- Anonymous user cannot change password.
- New password is required.
- Previous password is required.
- Previous password does not match existing password.

**401** [Unauthorized]

- User is not authorized to make call.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

## 3.9 Password Recovery

When a previously authenticated user used the system, they may need to recover their password. The request will be valid for one hour.

They will first need to request a forgot password before they are able to reset it.

Depending on your verification flow, whether it be email, text or security questions the user will need to either provide a code or answer to question to prove their knowledge of the request.

### 3.9.1 Forgetting Password

Creates a request for password reset that must have the matching data to reset to ensure request parity.

If using security questions, you can provide an attempt number to select which question is used for verification.

The attempt will load the question into the hint field to be asked of the user.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);

await client.ForgotPasswordAsync(username, attempt);
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**username** [string, required] Unique identifier for user or device.

**attempt** [integer, required] Identifies how many times a request has been made.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

await client.forgotPassword(username, attempt);
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**username** [string, required] Unique identifier for user or device.

**attempt** [integer, required] Identifies how many times a request has been made.

REST

```
POST https://api.meshydb.com/{accountName}/users/forgotpassword HTTP/1.1
Content-Type: application/json

  {
    "username": "username_testermctesterson",
    "attempt": 1
  }
```

**accountName** [string, required] Indicates which account you are connecting to.

**username** [string, required] Unique identifier for user or device.

**attempt** [integer, required] Identifies how many times a request has been made.

### Responses

**200** [OK]

- Generates forgot password response to be used for password reset.

Example Result

```
{
        "username": "username_testermctesterson",
        "attempt": 1,
        "hash": "...",
        "expires": "1900-01-01T00:00:00.000Z",
        "hint": "xxxx"
}
```

**400** [Bad request]

- Username is required.
- Anonymous user cannot recover password.

**404** [Not Found]

- User was not found.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

## 3.9.2 Check Hash

Optionally, before the user's password is reset you can check if the verification code, they provide is valid.

This would allow a user to verify their code before requiring a reset.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);

var check = new UserVerificationCheck();

var isValid = await client.CheckHashAsync(check);
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**username** [string, required] Unique identifier for user or device.

**attempt** [integer, required] Identifies which attempt hash was generated against.

**hash** [string, required] Generated hash from verification request.

**expires** [date, required] Identifies when the request expires.

**hint** [string, required] Hint for verification code was generated.

**verificationCode** [string, required] Value to verify against verification request.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

await client.checkHash({
                            username: username,
                            attempt: attempt:
                            hash: hash,
                            expires: expires,
                            hint: hint,
                            verificationCode: verificationCode
                        });
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**username** [string, required] Unique identifier for user or device.

**attempt** [integer, required] Identifies which attempt hash was generated against.

**hash** [string, required] Generated hash from verification request.

**expires** [date, required] Identifies when the request expires.

**hint** [string, required] Hint for verification code was generated.

**verificationCode** [string, required] Value to verify against verification request.

REST

```
POST https://api.meshydb.com/{accountName}/users/checkhash HTTP/1.1
Content-Type: application/json

  {
     "username": "username_testermctesterson",
     "attempt": 1,
     "hash": "...",
     "expires": "1/1/1900",
     "hint": "...",
     "verificationCode": "...",
  }
```

**accountName** [string, required] Indicates which account you are connecting to.

**username** [string, required] Unique identifier for user or device.

**attempt** [integer, required] Identifies which attempt hash was generated against.

**hash** [string, required] Generated hash from verification request.

**expires** [date, required] Identifies when the request expires.

**hint** [string, required] Hint for verification code was generated.

**verificationCode** [string, required] Value to verify against verification request.

### Responses

**200** [OK]

- Identifies if hash with verification code is valid.

Example Result

```
true
```

**400** [Bad request]

- Username is required.
- Hash is required.
- Expires is required.
- Verification code is required.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

### 3.9.3 Resetting Password

Take result from forgot password and application verification code generated from email/text or security question answer, along with a new password to be used for login.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);

await client.ResetPasswordAsync(resetHash, newPassword);
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**username** [string, required] User name that is being reset.

**expires** [date, required] Defines when hash will expire before it needs to be regenerated.

**hash** [string, required] Hash result of forgot password to verify request for password reset.

**newPassword** [string, required] New user secret credentials for login.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var passwordResetHash = await client.forgotPassword(username);

await client.resetPassword(passwordResetHash, newPassword)
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**username** [string, required] User name that is being reset.

**expires** [date, required] Defines when hash will expire before it needs to be regenerated.

**hash** [string, required] Hash result of forgot password to verify request for password reset.

**newPassword** [string, required] New user secret credentials for login.

REST

```
POST https://api.meshydb.com/{accountName}/users/resetpassword  HTTP/1.1
Content-Type: application/json

  {
    "username": "username_testermctesterson",
    "expires": "1-1-2019",
    "hash": "randomlygeneratedhash",
    "newPassword": "newPassword"
  }
```

**accountName** [string, required] Indicates which account you are connecting to.

**username** [string, required] Unique identifier for user or device.

**expires** [date, required] Defines when hash will expire before it needs to be regenerated.

**hash** [string, required] Hash result of forgot password to verify request for password reset.

**newPassword** [string, required] New user secret credentials for login.

### Responses

**204** [No Content]

- Identifies password reset is successful.

**400** [Bad request]

- Username is required.
- Hash is required.
- Expires is required.
- Verification code is required.
- Hash is expired.
- New password is required.
- Anonymous user cannot be reset.
- User has already been verified.
- Request hash is invalid.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

## 3.10 Logging Out

Log authenticated user out.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginAnonymouslyAsync(username);

await connection.SignoutAsync();
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**username** [string, required] Unique identifier for user or device.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var anonymousUser = await client.registerAnonymousUser();

var meshyConnection = await client.loginAnonymously(anonymousUser.username);

await meshyConnection.signout();
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**username** [string, required] Unique identifier for user or device.

REST

```
POST https://api.meshydb.com/{accountName}/connect/revocation HTTP/1.1
Content-Type: application/x-www-form-urlencoded

  token={refresh_token}&
  token_type_hint=refresh_token&
  client_id={publicKey}

(Form-encoding removed, and line breaks added for readability)
```

**accountName** [string, required] Indicates which account you are connecting to.

**refresh_token** [string, required] Refresh token identifying authorization with MeshyDB requested during Generating Token.

**publicKey** [string, required] Public identifier of connecting service.

### Responses

**200** [OK]

- Identifies successful logout.

**400** [Bad request]

- Invalid client id.
- Token is missing.
- Unsupported Token type.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

## 3.11 Creating Data

Create new custom mesh data into specified mesh name.

C#

```
// Mesh is derived from class name
public class Person: MeshData
{
  public string FirstName { get; set; }
  public string LastName { get; set; }
}

var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginAnonymouslyAsync(username);

var person = await connection.Meshes.CreateAsync(new Person(){
  FirstName="Bob",
  LastName="Bobberson"
});
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**username** [string, required] Unique identifier for user or device.

**mesh** [string, default: class name] Identifies name of mesh collection. e.g. person.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var anonymousUser = await client.registerAnonymousUser();

var meshyConnection = await client.loginAnonymously(anonymousUser.username);

var createdMesh = await meshyConnection.meshes.create(meshName,
                                                      {
                                                         firstName:"Bob",
                                                         lastName:"Bobberson"
                                                      });
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**username** [string, required] Unique identifier for user or device.

**meshName** [string, required] Identifies name of mesh collection. e.g. person.

REST

```
POST https://api.meshydb.com/{accountName}/meshes/{mesh} HTTP/1.1
Authentication: Bearer {access_token}
Content-Type: application/json

   {
      "firstName": "Bob",
      "lastName": "Bobberson"
   }
```

**accountName** [string, required] Indicates which account you are connecting to.

**access_token** [string, required] Token identifying authorization with MeshyDB requested during Generating Token.

**mesh** [string, required] Identifies name of mesh collection. e.g. person.

### Responses

**201** [Created]

- Result of newly created mesh data.

Example Result

```
{
   "_id":"5c78cc81dd870827a8e7b6c4",
   "firstName": "Bob",
   "lastName": "Bobberson"
}
```

**400** [Bad request]

- Mesh name is invalid and must be alpha characters only.
- Mesh property cannot begin with '$' or contain '.'.

**401** [Unauthorized]

- User is not authorized to make call.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

## 3.12 Retrieving Data

Retrieve single item from Mesh collection.

C#

```
// Mesh is derived from class name
public class Person: MeshData
{
  public string FirstName { get; set; }
  public string LastName { get; set; }
}

var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginAnonymouslyAsync(username);

var person = await connection.Meshes.GetData<Person>(id);
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**username** [string, required] Unique identifier for user or device.

**mesh** [string, default: class name] Identifies name of mesh collection. e.g. person.

**id** [string, required] Identifies location of what Mesh data to retrieve.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var anonymousUser = await client.registerAnonymousUser();

var meshyConnection = await client.loginAnonymously(anonymousUser.username);

var meshData = await meshyConnection.meshes.get(meshName, id);
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**username** [string, required] Unique identifier for user or device.

**meshName** [string, required] Identifies name of mesh collection. e.g. person.

**id** [string, required] Identifies location of what Mesh data to retrieve.

REST

```
GET https://api.meshydb.com/{accountName}/meshes/{mesh}/{id} HTTP/1.1
Authentication: Bearer {access_token}
Content-Type: application/json

      {
         "firstName": "Bob",
         "lastName": "Bobberson"
      }
```

**accountName** [string, required] Indicates which account you are connecting to.

**access_token** [string, required] Token identifying authorization with MeshyDB requested during Generating Token.

**mesh** [string, required] Identifies name of mesh collection. e.g. person.

**id** [string, required] Identifies location of what Mesh data to retrieve.

### Responses

**200** [OK]

- Mesh data found with given identifier.

Example Result

```
{
   "_id":"5c78cc81dd870827a8e7b6c4",
   "firstName": "Bob",
   "lastName": "Bobberson"
}
```

**400** [Bad request]

- Mesh name is invalid and must be alpha characters only.

**401** [Unauthorized]

- User is not authorized to make call.

**404** [Not Found]

- Mesh data was not found.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

## 3.13 Updating Data

Update Mesh data in collection by id.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginAnonymouslyAsync(username);
var person = await connection.Meshes.GetDataAsync<Person>(id);

person.FirstName = "Bobbo";

person = await connection.Meshes.UpdateAsync(person);
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**username** [string, required] Unique identifier for user or device.

**mesh** [string, required, default: class name] Identifies name of mesh collection. e.g. person.

**id** [string, required] Identifies unique record of Mesh data to replace.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var anonymousUser = await client.registerAnonymousUser();

var meshyConnection = await client.loginAnonymously(anonymousUser.username);

var meshData = await meshyConnection.meshes.update(meshName,
                                                   {
                                                      firstName:"Bob",
                                                      lastName:"Bobberson"
                                                   },
                                                   id);
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**username** [string, required] Unique identifier for user or device.

**meshName** [string, required] Identifies name of mesh collection. e.g. person.

**id** [string, required] Identifies unique record of Mesh data to replace.

REST

```
PUT https://api.meshydb.com/{accountName}/meshes/{mesh}/{id}  HTTP/1.1
Authentication: Bearer {access_token}
Content-Type: application/json

{
```

(continues on next page)

(continued from previous page)

```
    "firstName": "Bobbo",
    "lastName": "Bobberson"
}
```

**accountName** [string, required] Indicates which account you are connecting to.

**access_token** [string, required] Token identifying authorization with MeshyDB requested during Generating Token.

**mesh** [string, required] Identifies name of mesh collection. e.g. person.

**id** [string, required] Identifies unique record of Mesh data to replace.

### Responses

**200** [OK]

- Result of updated mesh data.

Example Result

```
{
    "_id":"5c78cc81dd870827a8e7b6c4",
    "firstName": "Bobbo",
    "lastName": "Bobberson"
}
```

**400** [Bad request]

- Mesh name is invalid and must be alpha characters only.
- Mesh property cannot begin with '$' or contain '.'.

**401** [Unauthorized]

- User is not authorized to make call.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

## 3.14 Searching Data

Filter Mesh data from collection based on query parameters.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginAnonymouslyAsync(username);

var pagedPersonResult = await connection.Meshes.SearchAsync<Person>(filter, page,␣
↪pageSize);
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**username** [string, required] Unique identifier for user or device.

**mesh** [string, required, default: class name] Identifies name of mesh collection. e.g. person.

**filter** [string] Criteria provided in a MongoDB format to limit results.

**orderby** [string] Defines which fields need to be sorted and direction in a MongoDB format.

**page** [integer, default: 1] Page number of results to bring back.

**pageSize** [integer, max: 200, default: 25] Number of results to bring back per page.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var anonymousUser = await client.registerAnonymousUser();

var meshyConnection = await client.loginAnonymously(anonymousUser.username);

var pagedResults = await meshyConnection.meshes.search(meshName,
                                                       {
                                                          filter: filter,
                                                          orderby: orderby,
                                                          pageNumber: page,
                                                          pageSize: pageSize
                                                       });
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**meshName** [string, required] Identifies name of mesh collection. e.g. person.

**username** [string] Unique identifier for user or device.

**filter** [string] Criteria provided in a MongoDB format to limit results.

**orderby** [string] Defines which fields need to be sorted and direction in a MongoDB format.

**page** [integer, default: 1] Page number of results to bring back.

**pageSize** [integer, max: 200, default: 25] Number of results to bring back per page.

REST

```
GET https://api.meshydb.com/{accountName}/meshes/{mesh}?filter={filter}&
                                                       orderby={orderby}&
                                                       page={page}&
                                                       pageSize={pageSize} HTTP/1.1
Authentication: Bearer {access_token}
```

(Line breaks added for readability)

**accountName** [string, required] Indicates which account you are connecting to.

**access_token** [string, required] Token identifying authorization with MeshyDB requested during Generating Token.

**mesh** [string, required] Identifies name of mesh collection. e.g. person.

**filter** [string] Criteria provided in a MongoDB format to limit results.

**orderby** [string] Defines which fields need to be sorted and direction in a MongoDB format.

**page** [integer, default: 1] Page number of results to bring back.

**pageSize** [integer, max: 200, default: 25] Number of results to bring back per page.

### Responses

**200** [OK]

- Mesh data found with given search criteria.

Example Result

```
{
   "page": 1,
   "pageSize": 25,
   "results": [{
                  "_id":"5c78cc81dd870827a8e7b6c4",
                  "firstName": "Bobbo",
                  "lastName": "Bobberson"
               }],
   "totalRecords": 1
}
```

**400** [Bad request]

- Mesh name is invalid and must be alpha characters only.
- Filter is in an invalid format. It must be in a valid Mongo DB format.
- Order by is in an invalid format. It must be in a valid Mongo DB format.

**401** [Unauthorized]

- User is not authorized to make call.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.

## 3.15 Deleting Data

Permanently remove Mesh data from collection.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginAnonymouslyAsync(username);

await connection.Meshes.DeleteAsync(person);
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**username** [string, required] Unique identifier for user or device.

**meshName** [string, required, default: class name] Identifies name of mesh collection. e.g. person.

**id** [string, required] Identifies unique record of Mesh data to remove.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var anonymousUser = await client.registerAnonymousUser();
```

(continues on next page)

(continued from previous page)

```
var meshyConnection = await client.loginAnonymously(anonymousUser.username);

await meshyConnection.meshes.delete(meshName, id);
```

**accountName** [string, required] Indicates which account you are connecting to.

**publicKey** [string, required] Public identifier of connecting service.

**username** [string, required] Unique identifier for user or device.

**mesh** [string, required] Identifies name of mesh collection. e.g. person.

**id** [string, required] Identifies unique record of Mesh data to remove.

REST

```
DELETE https://api.meshydb.com/{accountName}/meshes/{mesh}/{id} HTTP/1.1
Authentication: Bearer {access_token}
```

**accountName** [string, required] Indicates which account you are connecting to.

**access_token** [string, required] Token identifying authorization with MeshyDB requested during Generating Token.

**mesh** [string, required] Identifies name of mesh collection. e.g. person.

**id** [string, required] Identifies unique record of Mesh data to remove.

### Responses

**204** [No Content]

- Mesh has been deleted successfully.

**400** [Bad request]

- Mesh name is invalid and must be alpha characters only.

**401** [Unauthorized]

- User is not authorized to make call.

**404** [Not Found]

- Mesh data was not found.

**429** [Too many request]

- You have have either hit your API or Database limit. Please review your account.